CLAIMS:

1. A floating point unit (FPU) which generates a correction signal and an inverted leading zero signal, comprising:

exponent logic, the exponent logic configured to generate an exponent value, a first incremented exponent value, and a second incremented exponent value;

exponent adjust and rounding logic configured to receive the exponent value, the first incremented exponent value, and the second incremented exponent value, and wherein the exponent adjust and rounding logic is further configured to add the inverted leading zero signal to the exponent output value, the first incremented exponent value, and the second incremented exponent value, thereby producing an exponent output value, a first incremented exponent output value, and a second incremented exponent output value.

2. The FPU of Claim 1, wherein the inverted leading zero signal comprises an indication of the number of leading zeroes of an output of a fraction adder.

3. The FPU of Claim 1, wherein the exponent adjust and rounding logic is configured to receive the correction signal, the correction signal employable to select either the exponent value, the first incremented exponent output value, or the second incremented exponent output value.

4. The exponent adjust and rounding logic of Claim 3, wherein the leading zero signal is generated by a leading zero anticipator, and the correction signal is generated by a leading zero anticipator corrector.

5. The FPU of Claim 1, wherein the exponent output value, the first incremented exponent output value and the second incremented exponent output value are expressed as (n+x) bit two's-complement numbers with an n-bit bias, wherein n is the exponent width in a format, and wherein x is a variable representing an integer number of 2 or larger.

6. The FPU of Claim 5, wherein the format comprises a IEEE 754-1985 Standard for binary floating point arithmetic.

7. The FPU of Claim 5, wherein the second incremented exponent value and the leading zero signal value are input into an equality comparator, thereby producing a signal indicating that the second incremented exponent output value is one count larger than the largest exponent allowed in a selected format.

8. The FPU of Claim 7, wherein the exponent output values are n+2 bit two's complement numbers with an n-bit bias, and wherein the two most significant bits of the second incremented exponent output value are input into a comparator.

9. The FPU of Claim 8, wherein the comparator is configured to determine whether the two most significant bits equal "01", the comparator further configured to produce an output signal which indicates whether the first incremented exponent output value overflows.

10. The FPU of Claim 5, wherein the sign bit of the exponent output value is employed as a signal indicating that the first incremented exponent output value could cause an underflow condition, and wherein the sign bit of the

first incremented exponent output value is employed as a signal indicating that the second incremented exponent output value could cause an underflow condition.

12. The exponent adjust and rounding logic of Claim 1, further configured to accept a special case signal indicating that the exponent output computed by the exponent adjust and round logic is to be replaced by a special exponent value.

12. The exponent logic of Claim 1, further comprising a 3-way compound adder configured to generate the exponent output value, the first incremented exponent output value and the second incremented exponent output value.

13. A method of calculating an exponent, comprising:

receiving a first, second and third exponent input value;

receiving a leading zero signal;

generating an exponent value, a first incremented exponent value and a second incremented exponent value;

inverting the leading zero signal;

combining the inverted leading zero signal with the exponent value, the first exponent value and the second exponent value;

producing an exponent output value, a first incremented exponent output value and a second incremented exponent output value; and

selecting among the exponent output value, the first incremented exponent output value, and the second incremented exponent output value.

14. The method of Claim 13, further comprising generating the leading zero signal by a leading zero anticipator.

15. The method of Claim 13, further comprising:

receiving a leading zero correction signal; and

selecting from the exponent output value, the first incremented exponent output value or the second incremented exponent output value as a function of the leading zero correction signal.

16. The method of Claim 13, further comprising representing the exponent output value, the first incremented exponent output value and the second incremented exponent output value as a biased two's-complement number.

17. The method of Claim 16, wherein the bias is a 8 bit bias for a single precision calculation, and a 11 bit bias for a double precision calculation.

18. The method of Claim 17, further comprising:

selecting the most significant bits of the second incremented exponent output value; and

determining whether the selected most significant bits of the second incremented exponent output value are a positive non-zero value.

19. The method of Claim 18, wherein the step of selecting the most significant bits comprises:

dropping the least significant eight bits of the second incremented exponent output value if the second incremented exponent output value is associated with single precision format;

dropping the least significant eleven bits of the second incremented exponent output value if the second incremented exponent output value is associated with double precision format; and

selecting the remaining digits as the most significant digits.

20. The method of Claim 19, further comprising employing the outcome of the determination as indicia of an overflow condition associated with the first incremented exponent output value.

21. The method of Claim 17, further comprising:

determining whether the exponent value is larger than a predefined maximum value by at least two;

determining whether the second incremented exponent output value equals the predefined maximum value plus one; and

combining the result of the two determinations through the employment of OR logic.

22. The method of Claim 21, wherein the predefined maximum value is 127 if the first and second incremented output values are associated with single precision numbers.

23. The method of Claim 21, wherein determining that the second incremented exponent output value equals the maximum predefined value plus 1, further comprises:

appending a bit with a value of one to the front of the leading zero signal;

comparing the appended leading zero signal and the second incremented exponent value;

if the appended leading zero signal and the second incremented exponent value are equal, signalling that the second incremented exponent output value equals the predefined maximum value plus one; and

if the appended leading zero signal and the second incremented exponent value are not equal, signalling that the second incremented exponent output value does not equal the predefined maximum value plus one.

24. The method of Claim 17, further comprising checking for an underflow condition of the second incremented exponent output through checking the sign bit of the first incremented exponent output value.

25. The method of Claim 17, further comprising checking for an underflow condition of the first incremented exponent output value through checking the sign bit of the exponent output value.

26. The method of Claim 13, further comprising

selecting between the exponent computed by the exponent adjust and a special exponent result value.

27. The method of Claim 13, further comprising employing a three way adder to compute the exponent output value and the first and second incremented exponent output values.

28. In a floating point unit (FPU) having a leading zero anticipator (LZA) and associated LZA error correction, and means for computing an exponent, a method of compensating for said error comprising:

generating an LZA correction signal;

generating a plurality of options for said exponent as a function of adding the exponent to at least two separate values; and

executing overflow and underflow checks on the plurality of options during the step of generating a plurality of options.

29. A computer program product for calculating an exponent, the computer program product having a medium with a computer program embodied thereon, the computer program comprising:

computer code for receiving a first, second and third exponent input value;

computer code for receiving a leading zero signal;

computer code for generating an exponent value, a first incremented exponent value and a second incremented exponent value;

computer code for inverting the leading zero signal;

computer code for combining the inverted leading zero signal with the exponent value, the first exponent value and the second exponent value;

computer code for producing an exponent output value, a first incremented exponent output value and a second incremented exponent output value; and

selecting among the exponent output value, the first incremented exponent output value, and the second incremented exponent output value.

30. A processor for calculating an exponent, the processor including a computer program comprising:

computer code for receiving a first, second and third exponent input value;

computer code for receiving a leading zero signal;

computer code for generating an exponent value, a first incremented exponent value and a second incremented exponent value;

computer code for inverting the leading zero signal;

computer code for combining the inverted leading zero signal with the exponent value, the first exponent value and the second exponent value;

computer code for producing an exponent output value, a first incremented exponent output value and a second incremented exponent output value; and

selecting among the exponent output value, the first incremented exponent output value, and the second incremented exponent output value.